VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015

Lukáš Vrba

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Vrba**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: E LINKX a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. MSc. Donald David Davendra, Ph.D.**

Konzultant bakalářské práce: Ing Roman Hrdý

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 4. dubna 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. dubna 2015 .....

Chtěl bych poděkovat všem ze společnosti E LINKX a.s. za vytvoření příjemného prostředí. Zejména bych chtěl poděkovat webovému oddělení za získání řady nových znalostí a zkušeností a také za jejich pozitivní přístup. Jmenovitě bych rád poděkoval Ing. Romanovi Hrdému za umožnění absolvovat tuto praxi a Ing. Radkovi Petříkovi za jeho vedení v období letních prázdnin.

Abstrakt

Tato bakalářská práce popisuje průběh odborné praxe ve společnosti E LINKX a.s. V první části popisuje odborné zaměření společnosti a pracovní zařazení studenta. V druhé části jsou vypsány úkoly vykonávané v rámci praxe. Popis řešení těchto úkolů je v následující části. Závěrem práce je shrnutí znalostí využitých při vykonávání této praxe a její celkové zhodnocení.

Klíčová slova: Bakalářská práce, odborná praxe, E LINKX a.s., C#, .NET Framework, CSS, webová aplikace, Windows služba

Abstract

This bachelor thesis describes the course of professional practice in the company E LINKX a.s. In the first part it describes professional specialization of company and job position of student. In the second part is the list of performed tasks within professional practice. Description of these tasks is in next part. In the end of this thesis is summary of knowledge used during the professional practice and overall evaluation.

Keywords: Bachelor thesis, proffesional practice, E LINKX a.x., C#, .NET Framework, CSS, web application, Windows service

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
CSS	–	Cascading Style Sheets
GUI	–	Graphical User Interface
LINQ	–	Language Integrated Query
MSDN	–	Microsoft Developer Network
MSMQ	–	Microsoft Message Queuing
NSS	–	Nejvyšší správní soud
SOA	–	Service Oriented Architecture
SQL	–	Structured Query Language
T-SQL	–	Transact-SQL
UML	–	Unified Modeling Language
USR	–	User Story
VS	–	Visual Studio
WCF	–	Windows Communication Foundation
XML	–	Extensible Markup Language

Obsah

1 Úvod **5**

2 Popis odborného zaměření firmy a pracovního zařazení **6**
2.1 Odborné zaměření 6
2.2 Popis pracovního zařazení 6

3 Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti **7**
3.1 Vývoj aplikace pro synchronizaci souborů a složek v počítačovém clusteru. 7
3.2 Úpravy produktu HelpDesk 8
3.3 Přepracování uživatelského rozhraní 8

4 Zvolený způsob řešení zadaných úkolů **9**
4.1 Vývoj aplikace pro synchronizaci souborů a složek v počítačovém clusteru. 9
4.2 Úpravy produktu HelpDesk 17
4.3 Přepracování uživatelského rozhraní 20

5 Teoretické a praktické znalosti a dovednosti získané v průběhu studia a uplatněné studentem v průběhu odborné praxe **21**

6 Znalosti či dovednosti scházející studentovi v průběhu odborné praxe **22**

7 Dosažené výsledky v průběhu odborné praxe a její zhodnocení **23**

8 Reference **24**

Seznam tabulek

1 Aplikace pro synchronizaci souborů - časová náročnost 7
2 Úprava produktu HelpDesk - časová náročnost 8
3 Přepracování uživatelského rozhraní - časová náročnost 8

Seznam obrázků

1 Třídní diagram-práce s configuračním souborem 10
2 Desktopový klient 12
3 Technická dokumentace 16
4 Webový klient - nastavení účtů 17
5 Úprava HelpDesku - Štítky 18
6 UserStory - wizzard 19
7 UserStory - detail 20

Seznam výpisů zdrojového kódu

1 Konfigurační soubor 11
2 Generování unikátních jmen 14

1 Úvod

Tato bakalářská práce popisuje mou praxi ve firmě. Pro praxi jsem si vybral společnost E LINKX a.s., která sídlí v Ostravě. Praxi ve firmě jsem si vybral jako bakalářskou práci hlavně díky konferenci, která se konala 16. května 2014 a na které prezentovala řada zajímavých firem. Na této konferenci prezentovala i společnost E LINKX, která mě zaujala a proto jsem se rozhodl poslat žádost o vykonání odborné praxe právě jim.

Společnost E LINKX mi umožnila rozšířit znalosti programování v jazyce C# a vývoje .NET aplikací. Dále mi umožnila podílet se na tvorbě zajímavých produktů a vyzkoušet si spolupráci v týmu, což jsou důležité prvky pro budoucí profesní život.

V rámci praxe jsem si první s dalšími studenty vyzkoušel vývoj aplikace pro synchronizování dat na počítačovém clusteru. V rámci tohoto vývoje jsme si vyzkoušeli vypracovat počáteční analýzu ze zadání a následnou prezentaci návrhu zákazníkovi. Následně jsme si vyzkoušeli rozdělit vývoj aplikace na jednotlivé části a vytvořit časový odhad pro každou z nich. Dále jsme měli možnost si vyzkoušet vývoj jedné aplikace ve více lidech, kdy nestačí pouze umět programovat, ale je potřeba být schopen komunikovat a spolupracovat s ostatními. Po dokončení aplikace jsme měli za úkol vypracovat technickou a uživatelskou dokumentaci a následnou prezentaci finálního produktu.

V dalším kroku jsem byl přiřazen do jiného týmu, který vypracovával stejné zadání a na základě jejich implementace rozšířit aplikaci o webového klienta. Díky tomu jsem měl možnost si vyzkoušet navázat na práci někoho jiného a podílet se na rozšiřování produktu, o kterém jsem měl jen základní informace.

Nakonec jsem byl zařazen mezi programátory zabývající se vývojem webových produktů a zde jsem si rozšířil své znalosti vyvíjení aplikací v .NET frameworku. Kromě toho jsem se zde blíže seznámil s prostředím MS SQL, skriptovacím jazykem JavaScript a kaskádovými styly.

2 Popis odborného zaměření firmy a pracovního zařazení

2.1 Odborné zaměření

E LINKX a.s. [1] je českou společností, která působí na trhu více než 10 let. Již od svého počátku je spolehlivým a flexibilním partnerem všem členům skupiny eD' system Group i mimo ni.

Společnost E LINKX a.s. je systémovým integrátorem. Tato pozice vychází z rozsáhlého portfolia vlastních aplikací, produktů třetích stran a strategických partnerství s významnými subjekty na trhu informačních technologií. Působí aktivně v oblasti vývoje SW pro internetový obchod, internetové nástavby, webhostingu a vývoje specializovaných informačních systémů.

2.2 Popis pracovního zařazení

Po přijímacím pohovoru jsem společně s dalšími studenty absolvoval v rámci odborné praxe kurz, ve kterém jsme se seznámili s jednotlivými kroky vývoje softwaru a vyzkoušeli si spolupráci v týmu.

Poté jsem byl zařazen do skupiny vyvíjející webové produkty na pozici programátora ASP.NET, kde jsem spolupracoval na úpravách stávajících a vývoji nových interních i externích webových produktů.

3 Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti

3.1 Vývoj aplikace pro synchronizaci souborů a složek v počítačovém clusteru.

Zadáním bylo vytvořit software, který bude udržovat synchronizaci souborů a složek mezi počítači v clusteru. Nastavení složek a souborů, které se mají synchronizovat by mělo být minimálně přes konfigurační soubor, ideálně přes GUI. Změny se musejí detekovat automaticky z hlavního člena clusteru, kde aplikace poběží. Při spuštění aplikace zkontroluje, zda se soubory a složky na serverech neliší. Dále bude udržovat historii souborů. Aplikace musí také kontrolovat soubory a složky na vedlejších členech a upozornit správce v případě, že došlo ke změně. Synchronizace se bude provádět v určitých časových intervalech s možností provést ihned, odložit, nebo zrušit.

Práce	Hodiny
Desktopový klient	29
Windows služba	26
Práce s konfiguračním souborem	15
WCF	23
Testování	10
Dokumentace	10
Webový klient	50
Celkem	163

Tabulka 1: Aplikace pro synchronizaci souborů - časová náročnost

3.2 Úpravy produktu HelpDesk

HelpDesk [2] je webová aplikace pro technickou podporu, která primárně slouží pro vytváření a správu jednotlivých požadavků. Součástí je i evidence práce, statistické výstupy a manažerské reporty, které poskytují přehled o řešení jednotlivých úkolů, vytíženosti zaměstnanců apod. Tento produkt není určen jen pro zákazníky, ale firma Elinkx ho využívá i jako interní, protože je schopna díky němu efektivně rozdělovat jednotlivé požadavky mezi zaměstnance. Na tomto produktu bylo většinou mým úkolem přidávat novou funkcionalitu.

Práce	Hodiny
Zapracování dovolené	35
Přidání štítků na požadavky	45
Implementace User Story	140
Celkem	220

Tabulka 2: Úprava produktu HelpDesk - časová náročnost

3.3 Přepracování uživatelského rozhraní

Úkolem bylo přepracovat uživatelské rozhraní již existujícího informačního systému pro Nejvyšší správní soud. Jedná se komplexní systém, který pokrývá celý proces práce soudního výkonu včetně veškeré podpory pro rozhodování soudců. Jeho součástí je také například podpora práce úseku správy, nebo podpora tvorby judikatury od jejího vzniku až po tvorbu sbírky.

Práce	Hodiny
Celkem	127

Tabulka 3: Přepracování uživatelského rozhraní - časová náročnost

4 Zvolený způsob řešení zadaných úkolů

4.1 Vývoj aplikace pro synchronizaci souborů a složek v počítačovém clusteru.

Po zadání úkolu jsme museli vypracovat analýzu. Na základě této analýzy jsme se rozhodli, že aplikace se bude skládat ze dvou částí.

První částí je windows služba [3], což je v operačním systému Microsoft Windows speciální program, který běží dlouhodobě a není v přímém kontaktu s uživatelem. Hlavním důvodem, proč jsme zvolili windows službu, je její schopnost běžet dlouhodobě a zároveň možnost zapínání po startu systému. Tato služba má za úkol se starat o hlídání nastavených složek, nebo souborů a jejich následnou synchronizaci. Dále o zálohování předešlých verzí a o hlídání, jestli nedošlo ke změně na vedlejších členech clusteru. Služba musí také vypisovat informace o jednotlivých operacích i o chybách do logu.

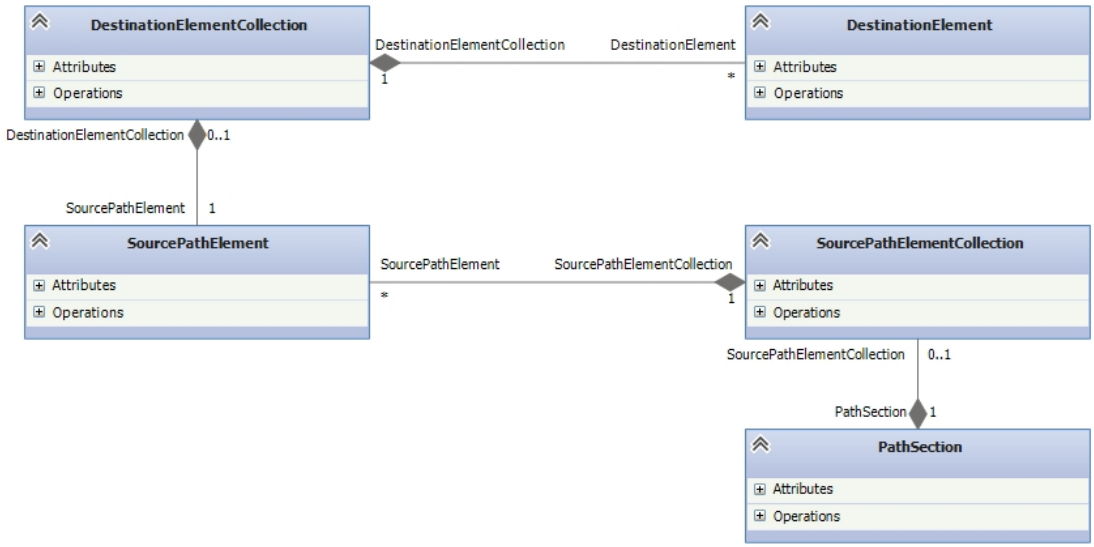
Druhou částí je desktopový (tlustý) klient. Hlavním úkolem tohoto klienta je možnost nastavování složek, nebo souborů, které má služba hlídat. Kromě toho klient informuje uživatele o nadcházející synchronizaci a umožní mu provést synchronizaci ihned, odložit ji, nebo zrušit. V případě zrušení synchronizace se nahradí změněný soubor, nebo složka původní verzí. Dále bude moci uživatel pomocí klienta nastavovat prodlevu, než dojde k synchronizaci, počet záloh a email, na který se má posílat případná notifikace.

Komunikace mezi službou a klientem je řešena pomocí WCF frameworku [4]. WCF sjednocuje dřívější technologie Microsoftu, které sloužily ke komunikaci. Mezi tyto technologie patří například ASP.NET Web Services, nebo MSMQ jejichž hlavní nevýhodou je, že vývoj aplikace je závislý na použité technologii. U WCF frameworku programátor nemusí při vývoji aplikace znát konkrétní programový model, ale stačí mu znát koncept WCF. Standart webových služeb, který WCF využívá, mu umožňuje vytvářet SOA aplikace, která definuje závislost přijímání a odesílání dat na webových službách. Díky tomu je vytvářena slabá závislost a díky tomu je možné se připojit z jakékoliv jiné platformy, která splňuje alespoň nezbytný kontrakt. WCF dále umožňuje například možnost odesílat zprávy podle různých vzorů jako request/reply, nebo one-way message.

Protože jsme pracovali ve třech na stejném projektu, tak jsme se dohodli nahrát projekt na GithHub, což je webová služba, která zjednodušuje vývoj softwaru v týmu.

4.1.1 Práce s konfiguračním souborem

Mým prvním úkolem bylo zjednodušit práci s konfiguračním souborem. Pro uložení nastavení ve stylu klíč, hodnota bylo možné použít sekci AppSettings. Problém nastal, když jsem chtěl do konfiguračního souboru ukládat hlídané a cílové cesty. Zde se ukázalo, že AppSettings není dostačující a že bude potřeba vytvořit v konfiguračním souboru novou sekci. Za tímto účelem jsem vytvořil třídy a každá z nich reprezentovala jednu úroveň v konfiguračním souboru. Diagram těchto tříd lze vidět na obrázku č. 1.



Obrázek 1: Třídní diagram-práce s configuračním souborem

Třídy DestinationElement a SourcePathElement reprezentují jednotlivé elementy v konfiguračním souboru a jako hodnotu mají nastavenou cestu k souboru, nebo složce. Třídy DestinationElementCollection a SourcePathElementCollection mají indexery, které odkazují na jednotlivé elementy. Třida PathSection k zabalení těchto tříd. Na výpisu č. 1 lze vidět výslednou strukturu konfiguračního souboru.

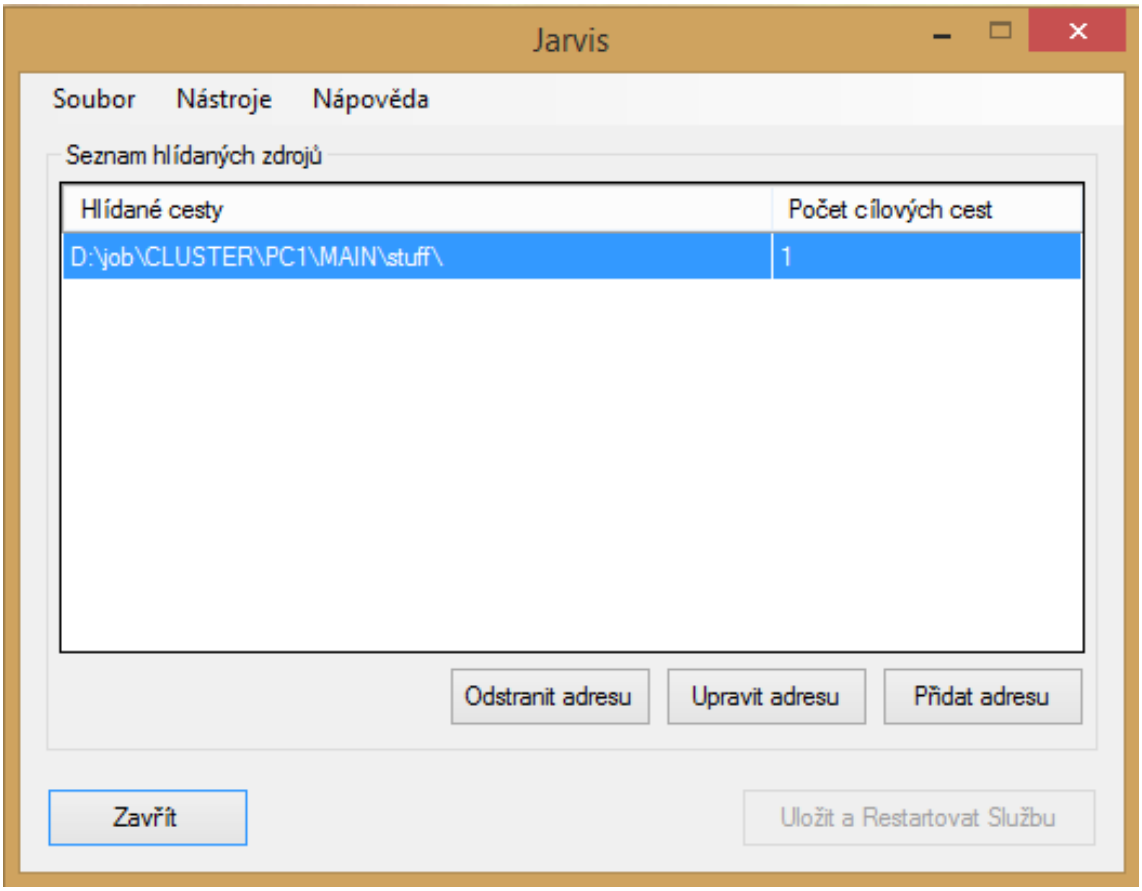
```
<PathSection>
    <Paths>
        <SourcePathElement sourcePath="D:\job\CLUSTER\PC1\stuff" backupName="stuff">
            <DestinationPaths>
                <DestinationElement destination="\\PC3\Documents\stuff" />
            </DestinationPaths>
        </SourcePathElement>
    </Paths>
</PathsSection>
```

Výpis 1: Konfigurační soubor

K těmto třídám jsem přidal statickou třídu ConfigControler, která slouží pro snadnější manipulaci se záznamy, které je potřeba načíst, nebo uložit do konfiguračního souboru. Z těchto tříd jsem následně vytvořil knihovnu, která se později implementovala pro službu i klienta, protože oba pracujou se stejným konfiguračním souborem.

4.1.2 Desktopový klient

Desktopového klienta jsem nevytvářel od začátku, ale přebral jsem ho po kolegovi, který jel na dovolenou. Klient ještě neměl téměř žádnou funkcionalitu, ale bylo to spíše interaktivní GUI. První věcí co mě zde zarazila bylo velké množství formů. Zjistil jsem, že řada těchto formů má stejné prvky a jediný rozdíl je v tom k čemu jsou určeny, protože kolega vytvářel zvlášť form pro vkládání nových záznamů a pro editaci. Proto jsem tyto formy sjednotil a jestli se má jednat o vkládání, nebo editaci rozhodovaly parametry v konstruktoru. Hlavní okno klienta lze vidět na obrázku 2.



Obrázek 2: Desktopový klient

V dalším kroku jsem implementoval dříve vytvořenou knihovnu pro práci s konfiguračním souborem a dodělal funkcionalitu jednotlivých částí klienta.

Doplňujícím úkolem bylo, že klient má fungovat ve dvou režimech a to s administrátorskými právy, nebo bez nich. V režimu s administrátorskými právy má uživatel plnou kontrolu nad klientem a může upravovat jednotlivá nastavení. V režimu bez administrátorských práv naopak klient funguje pouze k zobrazování notifikací o operacích prováděných službou. K dosažení toho úkolu jsem první musel rozchodit klienta v system tray. To bylo ve výsledku k mému překvapení jednoduchou záležitostí, hlavně díky Visual studiu. Následně jsem implementoval kontrolu, jestli byl program spuštěn s administrátorskými právy pomocí třídy WindowsPrincipal a pokud se ukázalo, že administrátorská práva nemá, tak se schová do system tray. Při požadavku na zobrazení okna si první vyžádá administrátorská práva a v případě, že je uživatel potvrdí, tak se restartuje.

Dále má klient za úkol hlídat, jestli je služba zapnutá, nebo vypnutá. K tomuto účelu byl využita třída ServiceController, díky které je možné ověřovat, v jakém stavu se daná služba právě nachází. Zde jsme to řešili způsobem opakovaného dotazování na stav, ale bylo nám řečeno, že to není efektivní. Z toho důvodu jsme využili metody WaitForStatus, která jako parametr přebírá enum, který definuje ruzné stavy, kterých služba může nabývat. Tato metoda se volá v odděleném vlákně, aby nedocházelo k zamrzání klienta v době, kdy čeká na změnu stavu.

4.1.3 Windows služba

Službu měl na starost kolega, který ji vytvořil a implementoval na ní FileSystemWatcher, pro hlídání zvolených složek a souborů. Při změně je replikoval na zvolenou destinaci. Já mu následně pomohl zpracovávat data získána z dříve vytvořené knihovny pro práci s konfiguračním souborem.

Služba byla ve stavu, kdy pouze hlídala změny a pokud k nějaké došlo, tak FileSystemWatcher zavolal nastavenou metodu a provedla se replikace. Mým úkolem bylo tyto metody předělat do stavu, kdy se replikace neprovádí okamžitě, ale po uplynutí daného časového intervalu. Díky prodlevě mezi provedenou změnou a nadcházející replikací jsem musel počítat s tím, že se provede více změn, než dojde k replikaci. Z toho důvodu jsem si vytvořil kolekci, která měla, jako klíč cestu k souboru ve kterém proběhla změna a jako hodnota byla reference na instanci třídy, která obsahovala další potřebné parametry. Při zachycení provedené změny se první zkontroluje, jestli už náhodou soubor není v kolekci. Pokud je, tak se ukončí, protože všechny záznamy v listu jsou určeny k replikaci a bylo by zbytečné dvakrát replikovat stejný soubor. Dalším krokem bylo nastavit kdy se má replikace provést. K řešení tohoto problému jsme využili třídy Timer, u které se nastavili událost na metodu, která se stará o replikaci a jako interval jsme nastavili hodnotu nastavenou v konfiguračním souboru. Po každé provedené změně se tento timer resetoval a začal znova odpočítávat. Po vypršení časového intervalu se zavolá nastavená metoda, která zkopíruje záznamy z kolekce, kterou následně vyprázdní a začne znova zachytávat změny. Následně se začaly procházet jednotlivé záznamy a u každého z nich

se provedla replikace. Pokud při pokus o replikování došlo k nějaké chybě, tak se zaznamenala zdrojová cesta, cílová cesta a zpráva proč se replikace nepovedla.

Když uživatel zvolí, že chce zrušit replikaci, tak služba musí umět nahradit změněné soubory, nebo složky původní verzí. Z tohoto důvodu vznikly dvě nové složky. Po startu služby se do složky COMMITTED překopírují všechny hlídané soubory a složky. Při provedení změny se upravené soubory překopírují do složky UNCOMITTED a v případě, že uživatel replikaci nezruší a ta v pořádku proběhne, tak se nahradí soubory ve složce COMMITTED verzí ze složky UNCOMITTED. Když uživatel zvolí, že chce změny zrušit, tak se smažou záznamy ve složce UNCOMITTED a změněné soubory a složky se nahradí verzí z COMMITTED. Po provedení replikace se provede záloha hlídané složky, nebo souboru, kvůli kterému replikace proběhla. V adresáři, který je nastaven v konfiguračním souboru, jako adresář pro zálohy se vytvoří složka s názvem hlídaného souboru, nebo složky. V této složce se vytváří adresáře, které obsahují zálohu hlídaných elementů. Tyto adresáře mají jméno ve formátu rrrr.MM.dd-hh.mm.ss, který reprezentuje čas, kdy daná záloha vznikla.

V případě funkce COMITTED i zálohy bylo potřeba vytvořit algoritmus, který by generoval unikátní jména. V konfiguračním souboru jsem vytvořil nový parametr, který udává unikátní jméno pro hlídaný soubor, nebo složku. Dále jsem vytvořil algoritmus pro generování tohoto unikátního jména. První jsem si načetl z konfiguračního souboru všechny prvky v sekci PathSection. Poté jsem u jednotlivých hlídaných cest zkontroloval, jestli mají nastavené unikátní jméno a pokud ne, tak jsem jim defaultně nastavil jako jméno název souboru, popřípadě složky, která se má hlídat. Následně se přešlo do cyklu while, který jde vidět ve výpisu č. 2. V tomto cyklu se první pomocí LINQ metody GroupBy sjednotí položky se stejným názvem do skupin a poté se tyto skupiny postupně předávají metodě, která má na starost vygenerování unikátního jména.

```
while(!allIsUnique) {
    var duplicate = elements.GroupBy(x => new { x.backupName }).Where(x => x.Any()).ToArray();

    allIsUnique = true;
    foreach(var dup in duplicate)
    {
        if (dup.Count() > 1)
        {
            allIsUnique = false;
            BackupUnique.CreateName(dup.ToList());
        }
    }
}
```

Výpis 2: Generování unikátních jmen

Aby jednotlivé názvy byly přehledné pro uživatele, tak v této metodě první rozdělím cestu k hlídanému souboru na jednotlivé části. Tyto části se poté postupně přidávají k názvu, dokud se nevytvoří unikátní název. Například pokud máme hlídaný soubor stuff.txt, který je ve složce Documents a hlídaný soubor stuff.txt, který je ve složce Downloads, tak výslednými názvy bude stuff.txt-Documents a stuff.txt-Downloads.

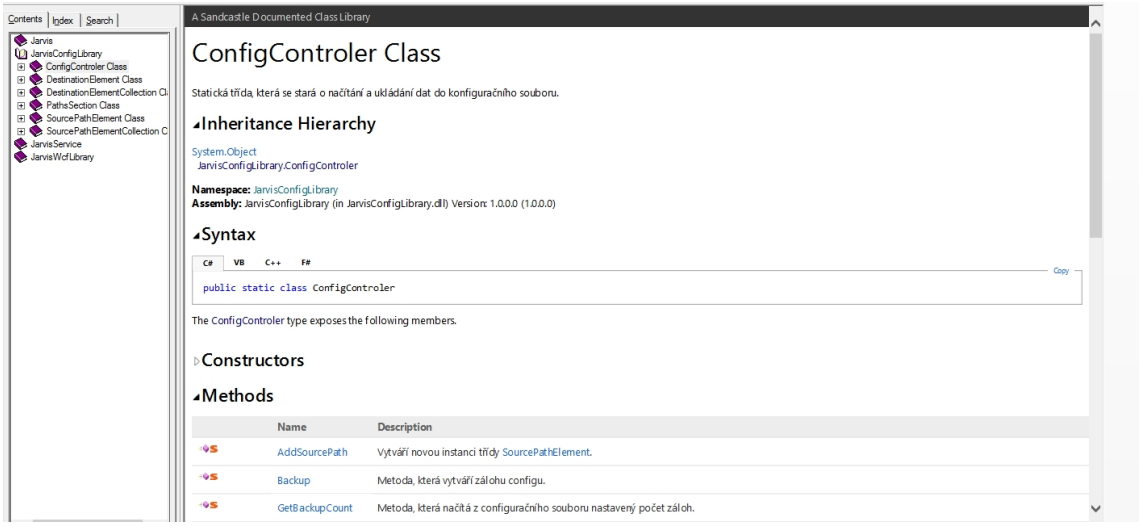
Služba musí dále umět zjišťovat, jestli se soubory, nebo složky na vedlejších členech neliší od toho na hlavním, což by znamenalo, že někdo měnil něco na vedlejších členech. V případě zjištění rozdílu služba odešle email pověřené osobě. Po spuštění tuto kontrolu provede služba u všech hlídaných složek a souborů a poté ji provádí před každou replikací pro položky, které se mají synchronizovat. V případě porovnávání před replikací není možné porovnávat s hlídaným členem, protože ten byl upraven a proto je jasné, že se nebudou shodovat a proto se porovnává s kopií, která je uložena ve složce COMMITTED. Na tomto úkolu jsem se podílel vytvořením metod, které porovnávají jednotlivé soubory a složky a jejich voláním. V případě složek se první načtou veškeré soubory ve složkách a pokud se neshoduje jejich výsledný počet, tak se zaznamená jako neshoda a není potřeba pokračovat. Pokud se jejich počet shoduje, tak se postupně načítají soubory z vedlejších členů a porovnávají se ze soubory z hlavního člena. Pokud se najde soubor, který má na vedlejším členu jiný název, ne na hlavním, tak to znamená neshodu, která se zaznamená a nemusí se hledat dál. Při porovnávání souborů se první zkontroluje, jestli se shoduje velikost. Pokud ano, tak se začnou soubory načítat po bytech a kontrolovat jestli se tyto byty shodují. V případě, že se neshodují, tak je stejně jako v předchozích případech zaznamenána neshoda do kolekce a metoda se ukončuje. Pokud po ukončení kontroly jsou v kolekci nějaké záznamy, tak se odešle zpráva na nastavený email s informacemi o neshodných souborech a složkách.

4.1.4 Komunikace mezi klientem a službou

Použití WCF frameworku nám bylo doporučeno školícím z firmy, který nám také ukázal základy konfigurace WCF. Na základě toho školení se mi podařilo navázat WCF komunikaci mezi klientem a službou. WCF běží na službě a klient se v pravidelných intervalech dotazuje, jestli nejsou žádné nové zprávy. Takto je to řešeno, protože klient může běžet na více účtech najednou, kdežto služba běží pod systémem a to znamená, že všichni uživatelé využívají stejnou službu. Pomocí této komunikace je klient schopen informovat uživatele o operacích prováděných službou, ale také ho může informovat za jak dlouho dojde k replikaci a umožnit uživateli zvolit, jestli chce replikaci provést ihned, odložit, nebo rovnou zrušit. Dále služba poskytuje klientovi pomocí WCF o nepovedených replikacích u kterých vypisuje i důvod proč se nepovedla.

4.1.5 Dokumentace

Po dokončení těchto částí jsme měli za úkol vypracovat uživatelskou a technickou dokumentaci. Pro technickou dokumentaci jsem zvolil program Sandcastle, který mi byl doporučen. Pro vygenerování dokumentace jsem první musel okomentovat všechny metody a proměnné, které jsem chtěl zdokumentovat a následně z těchto komentářů vytvořit pomocí VS XML dokument. Výsledný dokument po vygenerování v Sandcastlu je podobný dokumentaci MSDN viz obrázek č. 3.



Obrázek 3: Technická dokumentace

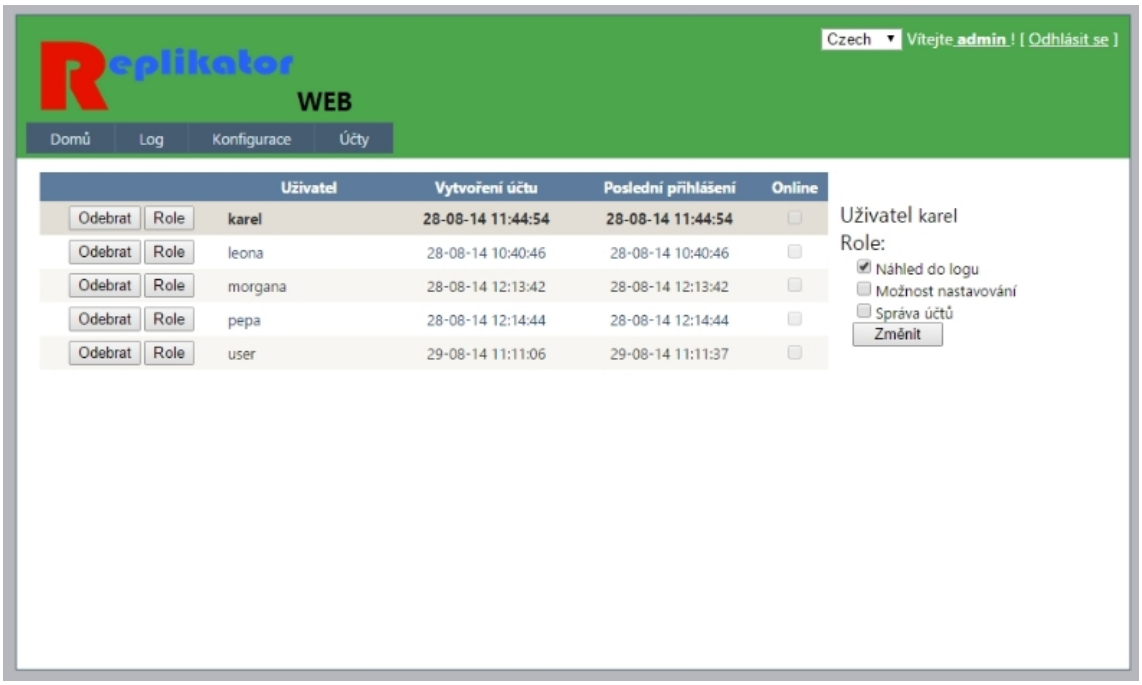
4.1.6 Webový klient

Po prezentaci finální aplikce jsem byl zařazen do druhého týmu, který dělal na stejném úkolu a na základě jejich řešení jsme museli vytvořit webového (tenkého) klienta. Zadáním bylo napsat webového klienta pomocí technologie ASP.NET. Tento klient má umět nastavovat stejné položky jako desktopový klient, ale nemusí řešit ovládání služby. Na druhou stranu musí umět získávat informace, které služba uložila do logu.

Pro vzhled jsme použili defaultní schéma VS 2010. Pro zpracovávání konfiguračních dat a získávání informací z logu, jsme museli vytvořit webovou službu, aby klient nemusel běžet na stejném serveru, jako aplikace pro synchronizování.

Webová služba pro načítání dat z logu využívá třídu EventLog díky které načítá všechny informace zalogované službou na synchronizaci souborů a složek. Pro načítání aktuálního nastavení a ukládání změn se využívá dll, které si pro práci s konfiguračním souborem napsala druhá skupina.

Dalším krokem, bylo vytvoření účtů, které mají práva na základě nastavených rolí. Za tímto účelem jsme použili nástroj ASP.NET Web Site Administration Tool, kde lze bez větších problémů tyto role a účty vytvořit. Následně jsem nastavil zobrazování různých položek menu a přístup na jednotlivé stránky podle těchto rolí. Za účelem snadnějšího spravování účtů jsem vytvořil stránky pro vytváření nových účtů, pro změnu hesla a pro upravování práv na účtu, popřípadě jejich odebrání viz obrázek č. 4.



Obrázek 4: Webový klient - nastavení účtů

4.2 Úpravy produktu HelpDesk

Ze začátku jsem dostával úkoly, které byly určeny spíše k tomu, abych se lépe seznámil s produktem HelpDesk a zejména abych se seznámil s používanými konvencemi ve firmě. Byly to úkoly typu přidat sloupec, do tabulky, nebo přidání nové položky do filtru.

4.2.1 Zapracování dovolené

Mým prvním větším úkolem bylo zpracování dovolené typem, uživatel bude nepřítomen od – do. V HelpDesku se jednotlivé dny dovolené, nemocenské a jiné důvody nepřítomnosti evidují po dnech, takže jsem musel tyto dny sloučit.

První se zkontroluje, jestli má v daný den volno. Pokud je ten den víkend, nebo státní svátek, tak se kontroluje první nadcházející pracovní den. V případě že daný den nemá volno, tak vím, že nemusím načítat z databáze starší záznamy.

Mimo to jsem musel do tohoto výsledku započítat i víkendy a svátky, protože když si zaměstnanec zaeviduje nepřítomnost například na pátek a na pondělí, tak to znamená, že je nepřítomen od pátku do pondělí.

4.2.2 Přidání štítků na požadavky

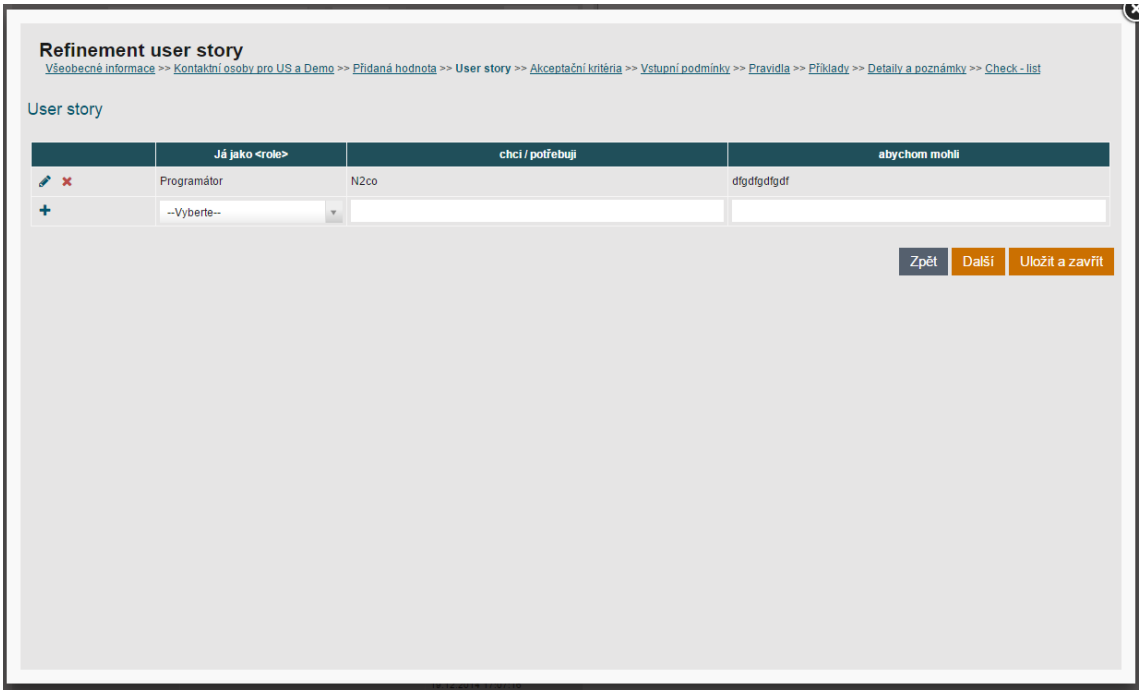
Zadáním bylo umožnit k jednotlivým požadavkům přidávat štítky, které jsou určeny hlavně pro lepší přehlednost a rychlejší vyhledávání mezi požadavky. Mezi seznamem štítků a seznamem požadavků bylo potřeba vytvořit „mezi tabulku“, protože jeden požadavek může mít více štítků a jeden štítek může mít více požadavků. Dále jsem do finálního řešení musel zaimplementovat i rozlišování mezi uživatelským a firemním štítkem, kde je rozdíl hlavně v tom, že firemní štítek vidí všichni uživatelé, ale uživatelské štítky jsou viditelné jen pro daného uživatele. Výsledný vzhled těchto štítků si můžete prohlédnout na obrázku č. 5.

Obrázek 5: Úprava HelpDesku - Štítky

4.2.3 Implementace User Story

User Story [5] se píše zejména v agilních a Scrum týmech. Hlavním důvodem proč se nevyužívají pouze požadavky, ale i User Story je způsob jakým se USR píše. Požadavek je většinou pouze nějaký výpis bodů, které nemusí být vždy jasné. USR má za účel vytvořit jasný obrázek, popřípadě příběh, protože lidský mozek daleko snadněji vnímá příběhy a obrázky, než technický popis v bodech.

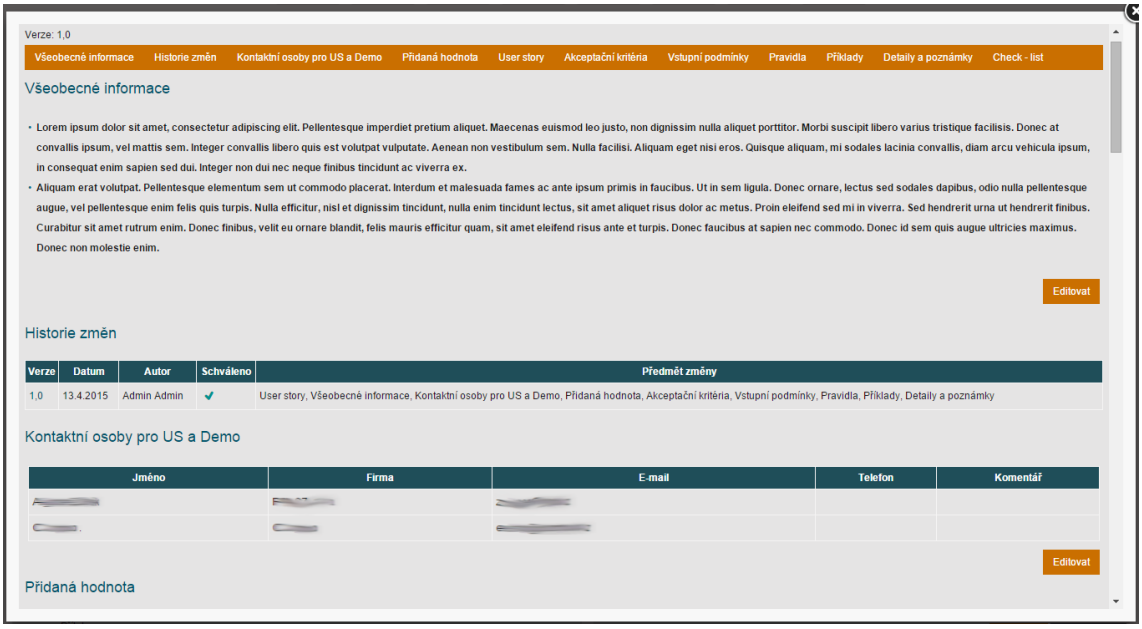
Dostal jsem za úkol navrhnout a následně implementovat toto User Story v elektronické podobě. Pro vytváření nového USR jsem implementoval wizzard, který jednotlivé položky v USR prochází krok za krokem. Řešení mělo umožňovat jednoduché přidávání nových kroků a měnit jejich pořadí bez většího zásahu do kódu. Za tímto účelem jsem si pro každý krok vytvořil zvlášť user control. O pořadí v jakém budou volány rozhoduje tabulka v databázi, takže pro přidání nového kroku do wizzardu stačí přidat záznam pro daný control s informací o jeho prioritě do tabulky. Výhody toho řešení jsem si mohl později i vyzkoušet, protože po dokončení úkolu mi bylo řečeno, abych přidal další 2 položky do USR. Finální výsledek tohoto wizzardu lze vidět na obrázku č. 6.



Obrázek 6: UserStory - wizzard

Následně bylo ale potřeba vytvořit výsledné USR. Zde se znova ukázaly výhody rozdělení jednotlivých korků, protože stačilo vytvořit pro jednotlivé controly mód náhledu

a editace. Poté jsem musel implementovat stránku, která už nevolala jednotlivé části krok po kroku, jako tomu bylo u wizzardu, ale naráz. Výsledný vzhled je vidět na obrázku č. 7



Obrázek 7: UserStory - detail

Z důvodu zachování předešlých verzí USR, se místo přepisování jedné verze, vytváří nová. K tomuto řešení bylo potřeba přidat možnost schvalování verzí, díky kterému se omezil jejich počet, protože dokud není verze schválena, tak se nevytváří nová, ale přepisuje stávající.

4.3 Přepracování uživatelského rozhraní

Staré uživatelské rozhraní informačního systému NSS bylo potřeba nahradit novým. Původně se nepoužíval css soubor, ale veškerá stylizace byla provedena přímo v jednotlivých prvcích. Bylo tedy potřeba odebrat původní stylizaci a nahradit ji daleko dynamičtější možností, kterou představuje css soubor. Ze začátku bylo potřeba přidat poměrně velké množství různých nastavení, ale ke konci už stačilo pouze nastavovat u jednotlivých prvků třídy, což ušetřilo velké množství práce a zároveň zajistilo jednotný vzhled napříč celým informačním systémem.

5 Teoretické a praktické znalosti a dovednosti získané v průběhu studia a uplatněné studentem v průběhu odborné praxe

V průběhu oborné praxe jsem využil většinu znalostí získaných během studia. Za zmínku stojí například znalost SQL, kterou jsem získal díky předmětu Úvod do databázových systému, a jeho rozšíření od společnosti Microsoft T-SQL se kterým jsem se seznámil v rámci Databázových a informačních systémů. V tomto předmětu jsem se poprvé setkal s technologií APS.NET, se kterou jsem se v rámci praxe často setkával. Dále mi hodně pomohl předmět Programovací jazyky 2, díky kterému jsem získal základní znalost jazyka C#. Při návrhu klienta mi pomohl předmět Uživatelské rozhraní ve kterém jsem se naučil vytvářet prostředí, které je pro uživatele přívětivé a snadno pochopitelné. Z předmětu Úvod do softwarového inženýrství jsem využil znalost jazyka UML. Pro získávání nových znalostí jsem většinou využíval anglické stránky, a proto za zmínku stojí i předmět Anglický jazyk, který mi pomohl rozšířit hlavně znalost odborné angličtiny.

6 Znalosti či dovednosti scházející studentovi v průběhu odborné praxe

V průběhu odborné praxe mi scházely hlubší znalosti jazyka C#. Mezi tyto znalosti patří například vývoje služeb, práce s konfiguračním souborem, nebo WCF komunikace. S většinou jsem se setkal v předmětu Architektura technologie .NET, který jsem ale absolvoval až v letním semestru 3. ročníku. Dalším předmětem, který by se mi hodil absolvovat o rok dříve byla Kompetence pro trh práce, kde jsem získal znalosti, které bych využil při vytváření životopisu, ústním pohovoru i následné spolupráci v týmu. Dále mi scházeli znalosti týkající se testování softwaru, popřípadě jeho analýzy a zejména zkušenosti vyvíjení jedné aplikace ve více lidech, kdy je hodně důležité porozumět algoritmům ostatních členů týmu.

7 Dosažené výsledky v průběhu odborné praxe a její zhodnocení

Díky této praxi jsem si rozšířil své znalosti programovacího jazyka C# a .NET technologií, kde jsem se podrobněji seznámil zejména s ASP.NET, které se ve škole věnovala malá pozornost. Oproti normální bakalářské práci jsem měl možnost spolupracovat v týmu zkušených programátorů a získat díky tomu řadu zkušeností, které budou hodně užitečné v dalším studiu i pozdějším profesním životě. Moje úkoly během praxe byly různorodé a díky tomu jsem měl možnost stále získávat nové zkušenosti a seznámil jsem se s řadou zajímavých řešení. Díky vzestupnému charakteru obtížnosti jednotlivých úkolů jsem byl schopen tyto úkoly zvládat a nestalo se mi, že by se mi nějaký ze zadaných úkolů nepovedlo vyřešit.

S praxí ve firmě jsem spokojen a jsem rád, že jsem si pro ni vybral společnost E LINKX a.s., která mi poskytla kvalitní pracovní prostředí, úžasný kolektiv a možnost podílet se na zajímavých produktech a úkolech díky kterým jsem získal řadu nových znalostí a zkušeností.

8 Reference

[1] O společnosti. *E LINKX a.s.* [online]. © 1999 – 2014 [cit. 2015-04-29].
Dostupné z: http://www.elinkx.cz/o-spolecnosti

[2] Technická podpora HelpDesk. *E LINKX a.s.* [online]. © 1999 – 2014 [cit. 2015-04-29].
Dostupné z: http://www.elinkx.cz/helpdesk

[3] Introduction to Windows Service Applications.
MSDN [online]. © 2015 [cit. 2015-04-29].
Dostupné z: https://msdn.microsoft.com/en-us/library/d56de412(v=vs.110).aspx

[4] What Is Windows Communication Foundation.
MSDN [online]. © 2015 [cit. 2015-04-29].
Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx

[5] User Stories: An Agile Introduction.
Agile Modeling [online]. © 2003-2014 [cit. 2015-04-27].
Dostupné z: http://www.agilemodeling.com/artifacts/userStory.htm